Docket No.: 87428.1520 **Patent** AF JFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES
APPEAL BRIEF FOR THE APPELLANTS
Ex parte Holger Nolte

In re Application of	:	Confirmation No.: 8873
Holger NOLTE et al.	:	Art Unit 2173
Serial No.: 10/001,940	:	Examiner: Raymond J. BAYERL
Filed: November 29, 2001	:	

For: SYSTEM AND METHOD FOR IMPLEMENTING A THREE-DIMENSIONAL GRAPHIC USER INTERFACE

BRIEF ON APPEAL

I. INTRODUCTION

This is an appeal from the final Office Action dated August 5, 2005. A Notice of Appeal was filed on December 5, 2005.

II. REAL PARTY IN INTEREST

The Real Party in Interest in the present application is Critical Reach, Inc. by way of an assignment.

III. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to the appellant, representative or assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

IV. STATUS OF THE CLAIMS

Claims 1-38 are pending in the application. Claims 1, 11, 22, 24, 29 and 33-38 are independent claims. Claim 2 depends on claim 1. Claim 3 depends on claim 1. Claim 4 depends on claim 1. Claim 5 depends on claim 1. Claim 6 depends on claim 5. Claim 7 depends on claim 6. Claim 8 depends on claim 6. Claim 9 depends on claim 7. Claim 10 depends on claim 4. Claim 12 depends on claim 11. Claim 13 depends on claim 11. Claim 14 depends on claim 11. Claim 15 depends on claim 11. Claim 16 depends on claim 11. Claim 17 depends on claim 16. Claim 18 depends on claim 17. Claim 19 depends on claim 17. Claim 20 depends on claim 17. Claim 21 depends on claim 20. Claim 23 depends on claim 22. Claim 25 depends on claim 24. Claim 26 depends on claim 24. Claim 27 depends on claim 26. Claim 28 depends on claim 27. Claim 30 depends on claim 29. Claim 31 depends on claim 30. Claim 32 depends on claim 30.

The Examiner rejects claim 1-38 under 35 U.S.C. §103(a) over Iwamura et al. (U.S. Patent No. 5,945,976) in view of Montgomery et al. (U.S. Patent No. 5,969,533).

V. STATUS OF THE AMENDMENTS

In response to a final Office Action dated August 5, 2005, an Amendment submitted on September 9, 2005 was entered amending claims 2, 12, 16 and 33 in order to comply with the Examiner's suggestions. An Advisory Action issued on September 21, 2005 indicates that the amendments were entered by the Examiner. A Notice of Appeal was filed on December 5, 2005.

VI. SUMMARY OF THE INVENTION

The present patent application is directed to software, systems and methods for the display of objects in a three-dimensional simulated environment as well as for the user-selection of items within the display.

Although computers and data communication networks have revolutionized the processes of exchanging information, they are primarily used to exchange one-dimensional or two-dimensional data. In other words, existing systems are efficient at communicating raw data such as text and numbers (i.e., one-dimensional data) and some formatted data such as graphs and text on a formatted page (i.e., two-dimensional data). However, three-dimensional data used in computer aided design (CAD) and computer aided manufacturing (CAM) tools tend to be handled by more advanced machines.

It is important to distinguish between three-dimensional display devices and three-dimensional graphical user interfaces. Many techniques for rendering images that appear to be three-dimensional are known. These systems use perspective drawing techniques to display three-dimensional images, but interact with users in a conventional two-dimensional fashion. **A three-dimensional graphical user interface, in contrast, requires that a user be able to select and interact with three-dimensional graphical objects, not just view them.**

Although various tools are being developed for three-dimensional rendering, these efforts are not directed to systems and methods that enable user interaction with three-dimensional graphic objects with programmatically defined behavior. As user-interactive systems must rely on complex mathematical computations to distinguish objects with in a particular window, they have not been widely accepted to implement three-dimensional graphical user interfaces.

Three-dimensional graphical displays are implemented on two-dimensional display devices where the image is rendered and drawn in a manner that visually conveys the three-dimensional relationships between objects. Using an x-y reference to indicate the horizontal and vertical axes of the display and a z axis to indicate distance from the viewer, at any given x-y location, any number of objects may exist in different z planes. Only one of the objects, however, is closest to the viewer.

Most graphical user interface (GUI) software for selecting displayed objects allows a user to identify a particular x-y location (or range of locations) using a mouse, keyboard, joystick, pen/tablet, or other user input device that allow a user to manipulate the position of a cursor on a display. In some instances the x-y location information is communicated from the GUI processes to other processes that provide information about the object(s) in the vicinity of a cursor. For example, information about a displayed object or a related help dialog may be displayed while a cursor hovers or floats above a particular x-y location. A user may select the object a that x-y location by activating a control on the user interface device. With two-dimensional data only a single object will exist at any selected x-y location.

However, a significant challenge for 3-D graphical user interfaces involves selection of the object closest to the viewer's perspective from amongst the layered objects that exist at a particular x-y location. The system must first determine which of the several objects at a particular x-y location is "nearest" to the user. GUI processes found in most operating systems communicate only x-y location information to external processes. In most cases, the GUI processes may communicate color information as well. **Hence, the information provided by the GUI processes is ambiguous with respect to which object is the closes to the user in a 3-D display**.

In three-dimensional graphical software, processes and data structures exist that maintain information about the object's placement in x, y and z dimensions. A "z buffer" refers to a software of hardware data structure that holds depth information for each given x-y location to manage which screen elements can be viewed and which are hidden behind other objects. A typical configuration of a z-buffer comprises a number of bits of memory associated with each display pixel where the bit value indicates the distance in the z-axis of the displayed pixel. A rendering algorithm can then draw only pixels with a z-buffer value indicating a closer position to the viewer (e.g., the pixels with the largest of smallest z-buffer value) at each x-y location. The z-buffer may be maintained by application software, or by specialized graph subsystem such as a graphics accelerator card in a personal computer. These subsystems accelerator card in a personal computer. These subsystems tend to be memory intensive to provide multiple bits of resolution in the z-axis for each pixel of the display.

When an application creates a 3D graphical object, a set of points is used to represent a shape. The points are then connected by lines to form a mesh or wireframe. The wireframe defines a set of polygons that represent surfaces of the graphical object. Once the polygons have been created, the application software can then shade the individual polygons to create the appearance of a solid object, or apply texture to the object. The greater the number of points used to define the shape, the greater the resolution that the object can be displayed.

In a complex scene comprising multiple objects, the application must create and track many objects and surfaces that may not be visible in a particular scene. Each time the display is drawn, application software selects the graphical objects having surfaces that will be visible on the screen at each pixel of the display. For each point in the scene, a surface depth ordering is performed by computing a z-distance for each object from a particular vantage point or "camera" view. The z-distance is typically expressed in some units of measure defined by the application itself such as "meters" or "feet" or an arbitrary scale used by the application. Intersections of multiple objects at each x-y location are detected, and one object to be displayed is selected for each pixel based on the z-distance (ignoring, for the moment, transparent objects).

Some of the rendering tasks may be handed off to a graphics subsystem, in which case the z-distance may be converted into a z-buffer value expressed in arbitrary units understood by the graphics subsystem. Information is communicated from the graphic subsystem to a two-dimensional display buffer in the form of x-y coordinates and color data for that coordinate.

Significantly, the display buffer, also called a "frame buffer" has no knowledge of the objects it is displaying or whether they are one-, two-, or three-dimensional.

To select a displayed object using a GUI a given x-y location is selected, which is used by the graphics subsystem to determine a "pick ray" extending from a virtual eye point used for rendering and the point a the x-y location selected. This ray is geometrically intersected with all polygons of all objects, which maybe up to many hundreds of thousands of polygons. The polygon with the nearest intersection to the viewer belongs to the selected object. This process is computationally expensive process.

Moreover, in client-server systems where the some or all of the processing must occur on a server that is physically distant from the display device, object selection processes are slow. Some improvements have been realized in the display of 3D graphical objects, but improvements that allow selection and user interaction with 3D graphical user interfaces has lagged. In essence, these systems portray a user interface that appears three-dimensional, but in practice is two-dimensional when in terms of object selection. Moreover, many proposed systems require specialized browser software or plug-ins to operate. As a result, implementation of lightweight computing devices to serve as three-dimensional graphical user interfaces has been impractical.

To remedy thee above-mentioned maladies, the inventors have devised a graphical user interface capable of rendering various three-dimensional graphical objects, where each pixel in the display has a separate color value in order to adequately differentiate various objects, as well as some form of object identification data stored with each pixel covered by the rendered image. Accordingly, as a user points to a particular pixel, the user can identify a particular graphical object located at the pixel.

VII. THE CLAIMED INVENTION

Independent Claim 1

A graphical user interface comprising:

a rendered image of at least one graphical object, wherein the graphical object uses a number of pixels on a display device;

a color value stored for each pixel in the display device; and

object identification data stored with each pixel covered by the rendered image, wherein the object identification data uniquely identifies the graphical object located at the pixel.

Independent Claim 11

A method for providing information to a program using a graphical user interface, the method comprising:

rendering an image of a plurality of graphical objects at specified locations of a two-dimensional display device;

storing a color value for each location in the two-dimensional display device; and storing object identification data for each of the specified locations, wherein the object identification data uniquely identifies one of the graphical objects at the at least one location.

Independent Claim 22

A computer-readable medium containing instructions for controlling a data processing system to perform a method of providing information to a program object, the method comprising:

rendering an image of a graphical object at specified locations of a two-dimensional display device;

storing a color value for each location in the two-dimensional display device;

storing object identification data for each of the specified locations, wherein the object identification data uniquely identifies one of the graphical objects at the at least one location; and

using the object identification data to identify the graphical objects.

Independent Claim 24

A method for associating graphical elements in a graphical user interface with metadata describing the graphical elements, the method comprising:

associating a unique object identification value with each graphical element;

providing a data structure having an entry for each graphical element, each entry being associated with a particular object identification value; and

storing metadata about an associated graphical element in a corresponding entry of the data structure.

Independent Claim 29

A computer program product embodied in a tangible medium comprising computer program devices configured to cause a computer to associate graphical elements in a graphical user interface with metadata describing the graphical elements, the computer program product comprising:

first program devices configured to cause the computer to associate a unique object identification value with each graphical element;

second program devices configured to cause the computer to implement a data structure having an entry associated with a particular object identification value; and

third program devices configured to cause the computer to store metadata about an associated graphical element in a corresponding entry of the data structure.

Independent Claim 33

A system for displaying and interacting with graphic objects, the system comprising:

a display device comprising a plurality of pixels arranged in a two-dimensional array, wherein graphical objects may be associated with any of the plurality of pixels;

a frame buffer having a plurality of entries where each entry is associated with one of the plurality of pixels;

object identification information corresponding to one of the graphical objects, the object identification information being stored in a frame buffer.

Independent Claim 34

A computerized system comprising:

a display comprising a plurality of pixels where each pixel is located at a defined coordinate;

an application process generating a plurality of graphical objects, each graphical object having a unique object identification (ID) value;

a rendering process receiving the plurality of graphical objects and determining visible surfaces of the plurality of graphical objects with respect to a predetermined perspective;

a pixel map comprising a plurality of entries, each entry corresponding to one of the visible surfaces determined by the rendering process;

object identification data associated with at least some of the entries in the pixel map;

a pointer movable to selected coordinates of the display, movement of which is controlled by a pointing device; and

an object identification process coupled to read the coordinates of a selected pixel pointed to by the pointer and extract the object identification information associated with the pixel.

Independent Claim 35

A user interface for a three-dimensional environment comprising:

a pointer, movement of which is controlled by a pointing device;

at least one graphical object displayed on at least one pixel of a display; and

a frame buffer having memory for the at least one pixel, the memory including a first field for holding display information and a second field for holding object identification information.

Independent Claim 36

A frame buffer comprising:

a plurality of memory locations, each memory being associated with a pixel of a display device;

display data stored the memory locations, the display data indicating how the pixel is to be activated; and

object identification data stored in at least some of the memory locations, the object identification data indicating a unique software graphical object being displayed by the pixel.

Independent Claim 37

A graphical user interface comprising:

a rendered image of at least one three-dimensional graphical object, wherein the graphical object uses a number of pixels on a display device, and wherein there is stored a respective color value for each pixel in the display device; and

object identification data stored with each pixel covered by the rendered image, wherein the object identification data uniquely identifies the graphical object located at the pixel.

Independent Claim 38

A graphical user interface comprising:

a rendered image of at least one three-dimensional graphical object, wherein the graphical object uses a number of pixels on a display device, and wherein the rendered image produced using a z-buffer technique; and

object identification data stored with each pixel covered by the rendered image, wherein the object identification data uniquely identifies the graphical object located at the pixel.

VIII. ISSUES

Whether claims 1-38 are not obvious under 35 U.S.C. §103(a) over Iwamura et al. (U.S. Patent No. 5,945,976) in view of Montgomery et al. (U.S. Patent No. 5,969,533).

IX. GROUPING OF CLAIMS

Each claim of this patent application is separately patentable, and upon issuance of a patent, will be entitled to a separate presumption of validity under 35 U.S.C. §282.

X. APPELLANT'S ARGUMENTS

A. **Relevant Description of the Applied Art**

Iwamura discloses a graphic data processing system that displays a simulated *three-dimensional scene* from a number of different perspectives using a vector map. See, Abstract and col. 1, lines 12-18. In an embodiment depicted in Fig. 5A and related text, the graphic data processing system can be used to point to a "ground object" via an indication cursor 501. In operation of the indication cursor 501, the graphic data processing system can use a number of approaches outlined in Figs. 9 and 10 and related text.

However, Iwamura makes no mention that any component of its graphics system uses either: (1) a color value stored for each pixel, or (2) object identification data stored with each pixel covered by a rendered image, issues admitterd by the Examiner on page 3 of the final Office Action. Thus, Iwamura does not teach or suggest each and every limitation as recited in the independent claims.

Montgomery discloses a method for selecting an item from a *two-dimensional* graphics screen. See, Abstract and Fig. 2. As illustrated in Fig. 2, a number of graphic objects 204 and 206 can displayed with each object having: (1) an "item identifier" that identifies each particular graphics object to be displayed, and (2) a "color number" that is the sole instrument that defines the color of a given graphics object. See, col. 3, line 64 to col. 4, line 5. As clearly shown by Fig. 2 and related text, each object is represented by a single color. That is, there is but a single color available for each object in the Montgomery disclosure. Accordingly, Montgomery does not teach, suggest or even appreciate the use of a device that allows a separate color value stored for each pixel. Thus, Montgomery does not provide for the deficiencies of Iwamura.

B. The Examiner has not established a *prima facie* case of obviousness under 35 U.S.C. § 103(a).

To establish a *prima facie* case of obviousness, (1) the prior art references must teach or suggest all the claim limitations, (2) there must be some motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify or combine the reference teachings, and (3) there must be a reasonable likelihood of success that the claimed combination will work. *All three requirement must be met.* See M.P.E.P. § 2143.

1. The Applied Art Does Not Teach or Suggest All the Claim Limitations.

As discussed above, none of the references discloses, suggests or even appreciates a graphical user interface that includes a rendered image of at least one graphical object, wherein the graphical object uses a number of pixels on a display device and a color value stored for each pixel in the display device, as recited in independent claim 1, and similarly recited in independent claims 11, 22, 24, 29 and 33-36.

(i) The Examiner has Misconstrued Claim Language.

Applicants respectfully assert that the Office Action has misinterpreted the claim limitation "*a color value stored for each pixel in the display device.*" While the final Office Action correctly states on page 7 that 'the word "separate" is not associated with "a color value of each pixel"', Applicants respectfully assert that the only interpretation feasible dictates a separate/respective color value for each pixel in a display device based both on the text of the specification and the plain meaning of the claim language.

For example, page 10, line 29 to page 11, line 2 of the present specification recites: "As currently implemented, most computer screens 201 comprise a plurality of locations (e.g., "pixels") arranged in a two-dimensional array. Each pixel is on or off in a pure monochrome system, or activated with a red-green-blue (RGB) or cyan-magenta-yellow (CMY) or the like to represent color on a pixel-by-pixel basis." {emphasis added}

Additionally, the Office Action's interpretation, as is exemplified on page 7 (e.g., "Montgomery has a pixel-by-pixel storage across an object's surface ...") indicates that the Office Action is attempting to interpret the language "*a color value stored for each pixel in the display device*" into meaning "*a color value stored for each pixel of a graphical object in the display device.*"

(ii) Even Assuming the Examiner's Claim Construction, Montgomery is insufficient to Establish a Prima Facie case of Obviousness.

Applicants do not contest that Montgomery stores a color value for each object. However, Applicants do point out that not every pixel in the Montgomery device is covered by an object, only objects within the display. See, e.g., Figure 7. Without such, Montgomery cannot not satisfy the limitation "*a color value stored for each pixel in the display device*".

2. The Examiner has not provided the requisite motivation under 35 U.S.C. § 103(a) to make the specific claimed combination.

Even assuming the Examiner has satisfied the "all elements" requirement discussed above, Applicants respectfully point out that the Examiner has failed to provide an appropriate motivation based on the cited references, or alternatively failed to show that such a motivation could be derived from the knowledge generally available to one of ordinary skill in the art to modify Iwamura using the teachings of Montgomery. The cited "motivations" made by the Examiner are insufficient as a matter of law in that the Examiner has failed to provide "some motivation, suggestion, or teaching of the desirability of making the specific combination that was made by the Applicants." In re Dance, 160 F.3d 1339, 1343, 48 USPQ2d 1635, 1637 (Fed. Cir. 1998). Some particular findings must be made as to the reason a skilled artisan having no knowledge of the claimed invention would have selected the specific components for combination in the manner claimed. In re Kotzab, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000).

For example, while the final Office Action states on pages 4 that '*it would have been obvious ... to use the item buffer technique of Montgomery to assign "identification data" to the scene image objects of Iwamura [in order] to provide the Iwamura user with a direct indexing to the identities of the contents of the scene image,*' this stated motivation is problematic for a number of reasons.

First, the stated motivation is not found in any reference of record, or supported by any basis or theory having support in the references of record.

Second, it is not apparent that the virtual reality simulator of Iwamura is remotely compatible with, or would benefit from, the item buffer described in Montgomery as compared to the present (z-buffering) system used by Iwamura.

A review of Iwamura and the underlying technology reveals that there can be NO motivation (or even a reasonable likelihood of success) to modify the three-dimensional system of Iwamura to use Montgomery's item buffer (which is designed with two-dimensional graphics rendering in mind) to derive an improved three-dimensional rendering system. As with most three-dimensional rendering systems, Iwamura uses a "z-buffer" imaging approach to determine whether a particular object will be obfuscated by a second object (see, e.g., col. 8, line 53 of Iwamura), whereas the imaging approach used in Montgomery is limited by its "item buffer" technique to obfuscate objects based on order within a buffer, i.e., subsequent items listed obfuscate earlier items. See, Fig. 2 and col. 4, line 9+ of Montgomery. Accordingly, the devices and methods of Montgomery are totally unsuitable for use with Iwamura, or in any system using modern three-dimensional graphics rendering. At best, a z-buffering system modified by Montgomery's item-buffer would be slower and would introduce rendering errors.

While the final Office Action replies to Applicants' earlier assertions about the total lack of suitability or need to modify a z-buffer or use Montgomery's item buffer in addition to a z-buffer stating that "*it is not true ... since the resultant output of such rendering is a two dimensional image, whose pixels are then directly handled by a pixel-by-pixel arrangement such as Montgomery's*", Applicants again assert that this statement has no basis in reality. Once a z-buffering technique is performed, there is absolutely no subsequent use for any item-buffer as the z-buffering process already renders each pixel observable from a given perspective on a two-dimensional display. It is a fundamental purpose of z-buffering. Accordingly, Montgomery's item buffer cannot enhance Iwamura.

Further, Montgomery's item buffer can not replace Iwamura's z-buffer as Montgomery's item buffer can not properly be used for a true three-dimensional scene where the perspective of a scene will change dynamically as: (1) Montgomery's item-buffer can only be used for two-dimensional objects (2) viewed from a single perspective.

Regarding issue (1), a review of Montgomery shows that only two-dimensional systems are described, and only two-dimensional objects, e.g., rectangles and triangles, are discussed. In fact, the term "three-dimensional" as well as any three-dimensional object, e.g. "cube" and "sphere", do not appear anywhere in the text of Montgomery.

Regarding issue (2), nowhere does Montgomery provide for any change in viewing perspective. For example, by changing a viewing perspective of a square from "head-on" to 60 degrees, a viewer could expect to see a rectangle (even ignoring shading, which is also not provided for by Montgomery). By further changing the viewing perspective by 90 degrees, a viewer would see a line. Where two separate two-dimensional objects are presented one in front of another, without depth information of each of the two objects (not provided for by Montgomery) it is impossible to determine the change in scene as a viewing perspective changes angularly, e.g, by rotation, or linearly, i.e., by approaching up to and even through one or more objects.

Assuming for a moment that Montgomery's item buffer were adapted for use in three-dimensions by arbitrarily assigning each point of each two-dimensional object three-dimensional coordinates, the process shown in Montgomery might be perhaps closest to a technique known as "*z-sorting*." The z-sorting algorithm simply displays all objects serially, starting with those objects furthest back (with the largest z-axis values). The z-sorting algorithm does not require a z-buffer, but it is slow and does not render intersecting objects correctly.[1]

Applicants respectfully assert that even a modest investigation will reveal that the z-buffering technique is one of the most efficient (if not the most efficient) three-dimensional rendering techniques known. Accordingly, Applicants suggest that any modification made to z-buffering (or the variant known as "w-buffering"), including that suggested by the Examiner, will result in a less efficient, slower and less capable system.

[1] Applicants respectfully request the Board carefully review z-buffering literature. Two websites currently available of interest (reproduced in the Appendix) include: http://www.webopedia.com/TERM/Z/Z_buffering.html and http://www.visionengineer.com/comp/z_buffering.shtml.

While the Examiner's Advisory Action states that "*[f]or each pixel in the finally-displayed image, Montgomery et al. assigns an item value ... is advantageous ... since this will allow Iwamura et al.'s objects to be properly reference, once they've been readied for the final 2-dimensional display*", this statement suffers from the same problems as the statements made in the various Office Actions in that there is nothing supplied from anything in any of the art of record – no supporting statements – no basis upon which to form a theory – nothing – in order to make this assertion to make the <u>specific</u> combination asserted in the various Office Actions. It is but a conclusory statement made without support from any of the art of record.

While the Advisory Action further states that the "*Examiner is not attempting a <u>literal combination of the **entire** buffering arrangement</u> of both Iwamura et al. and Montgomery et al., but the teachings of the concepts of 3-dimensional rendering (Iwamura et al.) and buffering (Montgomery et al.), as they are applicable to these broad claims,*" {emphasis added}, the Examiner is trying to make <u>some form of literal combination</u> of *Iwamura and Montgomery* (that of the claim language), and the problem that arises is that the combination cán't literally be done, make no meaningful sense or are ultimately undesirable. One cannot functionally append an item buffer to a z-buffer - it makes no sense technically and serves no purpose. One cannot replace a z-buffer with an item buffer as item buffers are not suited to three dimensional rendering. One cannot conceptually modify a z-buffer as asserted by the Examiner without destroying essential functionality and compromising speed and efficiency.

But even ignoring all this, the Examiner still provides no basis of support to replace/modify the buffering system of Iwamura with the buffering system of Montgomery using anything but conclusory statements that are but than an attempt at hindsight recognition using the Applicants' specification as a guide. Iwamura already has a buffering system known as "z-buffering". Why one of ordinary skill in the art would alter this well-known and highly efficient buffering scheme (that is the best system presently known in the art) to create something new and to all appearance something less efficient or totally unworkable is highly problematic.

While the Advisory Action goes on to state that "*it does not matter that the z-buffering may have been employed ... since the result of image data in two dimensions that then benefits from the usage of Montgomery et al.'s joint buffering of color and items – regardless of Montgomery et al's possible suggestion as to its own form of initial buffering*," Applicants respectfully object to the Examiner's insinuations that that z-buffering is merely a "possibly suggestion" of Iwamura. The term "z buffer" literally appears in Iwamura. See, e.g., col. 8, line 53 of Iwamura. Add to this the near universal use and known utility of z-buffering and it is apparent that z-buffering is more than but "possible suggestion" by Iwamura.

Further, it <u>does matter</u> that z-buffering is employed in Iwamura as z-buffering is the best known approach to rendering three-dimensional graphics known to those of ordinary skill in the art. It is beyond any remote possibility that the Examiner has devised an advantageous rendering scheme (compared to z-buffering) in his attempt to form a motivation to combine two unrelated references in order to assert an obviousness rejection to the present claims. The Examiner's suggested combinations of Iwamura and Montgomery are technically undesirable if not totally infeasible. Even if feasible and somehow desirable, the Examiner has not provided the requisite motivation necessary to make the specific combination necessary to establish a *prima facie* case of obviousness.

3. <u>There is no reasonable likelihood of success that the Examiner's claimed combination will work</u>

As mentioned above, one cannot functionally append an item buffer to a z-buffer - it makes no sense technically and serves no purpose. One cannot replace a z-buffer with an item buffer as item buffers are not suited to three dimensional rendering. One cannot conceptually modify a z-buffer as asserted by the Examiner without destroying essential functionality and compromising speed and efficiency. Accordingly, there can be no reasonable likelihood of success that the Examiner's claimed combination will work.

XI. CONCLUSION

For all of the above-noted reasons, it is strongly contended that certain, clear and important distinctions exist between the present invention as recited in claims 1-28 and the cited references as provided in the Examiner. It is further contended that these distinctions are more than sufficient to render the claimed invention unobvious to a person of ordinary skill in the art at the time the invention was made.

This final rejection being in error, therefore, it is respectfully requested that this Honorable Board of Patent Appeals and Interferences reverse the Examiner's decision in this case, and indicate the allowability of claims 1-38.

In the event that this paper is not timely filed, Applicants respectfully petition for an appropriate extension of time. Please charge any fee deficiencies or credit any overpayments to Deposit Account No. 50-2036.

Respectfully submitted,

BAKER & HOSTETLER LLP

[illegible]

B. Y. Mathis
Reg. No. 44,907

Attachment: Appendix

Washington Square, Suite 1100
1050 Connecticut Avenue, N.W.
Washington, D.C. 20036
Phone: (202) 861-1500
Fax: (202) 861-1783
Date: December 19, 2005

APPENDIX 1

IN THE CLAIMS:

1. A graphical user interface comprising:

a rendered image of at least one graphical object, wherein the graphical object uses a number of pixels on a display device;

a color value stored for each pixel in the display device; and

object identification data stored with each pixel covered by the rendered image, wherein the object identification data uniquely identifies the graphical object located at the pixel.

2. The graphical user interface of claim 1, wherein the rendered image comprises a three-dimensional image comprising at least two graphical objects that are co-located at at least one point in an x-y view plane, wherein only one of the co-located graphical objects is visible at a pixel corresponding to the at least one point and the object identification data for the pixel identifies the visible graphical object.

3. The graphical user interface of claim 1 further comprising processes for writing the object identification data with the color value to a frame buffer when drawing the rendered image of the at least one graphical object on the display device.

4. The graphical user interface of claim 1 further comprising processes for reading the object identification data with the color value from a frame buffer.

5. The graphical user interface of claim 1 further comprising a cursor pointing to a particular coordinate of the graphical display device, movement of which is controlled by user input.

6. The graphical user interface of claim 5 further comprising processes for reading the object identification data. with the color value from the frame buffer of a pixel associated with the particular coordinate pointed to by the cursor.

7. The graphical user interface of claim 6 further comprising an interface for exporting the object identification information to an external process.

8. The graphical user interface of claim 6 further comprising a metadata information display object displayed on the display in response to the object identification data of the pixel associated with the particular coordinate pointed to by the cursor.

9. The graphical user interface of claim 6 further comprising an object-identification value indexed data structure holding metadata about the an object uniquely identified by the object identification value.

10. The graphical user interface of claim 4 further comprising processes for reading the object identification data with the color value from a frame buffer.

11. A method for providing information to a program using a graphical user interface, the method comprising:

rendering an image of a plurality of graphical objects at specified locations of a two-dimensional display device;

storing a color value for each location in the two-dimensional display device; and storing object identification data for each of the specified locations, wherein the object identification data uniquely identifies one of the graphical objects at the at least one location.

12. The method of claim 11 wherein from a particular viewpoint, at least two of the graphical objects intersect at at least one intersection location of image, and the object identification data stored at the pixel associated with the intersection location identifies an object that is visually closest to the viewpoint.

13. The method of claim 11 further comprising:

using the object identification data to identify one of the intersecting graphical objects.

14. The method of claim 11 further comprising writing the object identification data with the color value to a frame buffer when drawing the rendered image of the plurality of graphical objects on the two-dimensional display device.

15. The method of claim 11 further comprising reading the object identification data with the color value from a frame buffer.

16. The method of claim 11 further comprising:

receiving user input indicating movement on the two-dimensional display device; and

pointing to a particular location of the two-dimensional graphical display device indicated by the user input.

17. The method of claim 16 further comprising reading, from the frame buffer, the object identification data with the color value of a pixel associated with the particular location pointed to.

18. The method of claim 17 further comprising exporting the object identification information to an external process.

19. The method of claim 17 further comprising displaying a metadata information display object on the two-dimensional display device in response to the object identification data read from the particular location pointed to.

20. The method of claim 17 further comprising storing metadata about the graphical objects in an object-identification value indexed data structure; and

using the object-identification value read from the particular location pointed at to index into the data structure so as to retrieve metadata about a particular object.

21. The method of claim 20 further comprising displaying a metadata information display object displayed on the display in response to the object identification data.

22. A computer-readable medium containing instructions for controlling a data processing system to perform a method of providing information to a program object, the method comprising:

rendering an image of a graphical object at specified locations of a two-dimensional display device;

storing a color value for each location in the two-dimensional display device;

storing object identification data for each of the specified locations, wherein the object identification data uniquely identifies one of the graphical objects at the at least one location; and

using the object identification data to identify the graphical objects.

23. The computer-readable medium of claim 22 wherein at least two of the graphical objects intersect at least one location on the two-dimensional display device.

24. A method for associating graphical elements in a graphical user interface with metadata describing the graphical elements, the method comprising:

associating a unique object identification value with each graphical element;

providing a data structure having an entry for each graphical element, each entry being associated with a particular object identification value; and

storing metadata about an associated graphical element in a corresponding entry of the data structure.

25. The method of claim 24 further comprising storing the unique object identification value in a frame buffer.

26. The method of claim 24 further comprising:

determining display information describing how each graphical element is to be displayed; and

storing the unique object identification value corresponding to the graphical element in a frame buffer location associated with the particular display location.

27. The method of claim 26 further comprising:

receiving user-input identifying a user-identified location in the display; and

extracting the object identification value associated with the user-identified location from the frame buffer.

28. The method of claim 27 further comprising retrieving the stored metadata using the extracted object identification value.

29. A computer program product embodied in a tangible medium comprising computer program devices configured to cause a computer to associate graphical elements in a graphical user interface with metadata describing the graphical elements, the computer program product comprising:

first program devices configured to cause the computer to associate a unique object identification value with each graphical element;

second program devices configured to cause the computer to implement a data structure having an entry associated with a particular object identification value; and

third program devices configured to cause the computer to store metadata about an associated graphical element in a corresponding entry of the data structure.

30. The computer program product of claim 29 further comprising:

fourth program devices configured to cause the computer to determine display information describing how each graphical element is to be displayed;

fifth program devices configured to cause the computer to select one graphical elements as a front-most graphical element at each pixel; and

sixth program devices configured to cause the computer to store the unique object identification value corresponding to the front-most graphical element in a frame buffer location associated with the particular display location at which the front-most graphical element is displayed.

31. The method of claim 30 further comprising:

seventh program devices configured to cause the computer to receive user-input identifying a user-identified location in the display; and

eighth program devices configured to cause the computer to extract the object identification value associated with the user-identified location from the frame buffer.

32. The method of claim 30 further comprising ninth program devices configured to cause the computer to retrieve the stored metadata using the extracted object identification value.

33. A system for displaying and interacting with graphic objects, the system comprising:

a display device comprising a plurality of pixels arranged in a two-dimensional array, wherein graphical objects may be associated with any of the plurality of pixels;

a frame buffer having a plurality of entries where each entry is associated with one of the plurality of pixels;

object identification information corresponding to one of the graphical objects, the object identification information being stored in a frame buffer.

34. A computerized system comprising:

a display comprising a plurality of pixels where each pixel is located at a defined coordinate;

an application process generating a plurality of graphical objects, each graphical object having a unique object identification (ID) value;

a rendering process receiving the plurality of graphical objects and determining visible surfaces of the plurality of graphical objects with respect to a predetermined perspective;

a pixel map comprising a plurality of entries, each entry corresponding to one of the visible surfaces determined by the rendering process;

object identification data associated with at least some of the entries in the pixel map;

a pointer movable to selected coordinates of the display, movement of which is controlled by a pointing device; and an object identification process coupled to read the coordinates of a selected pixel pointed to by the pointer and extract the object identification information associated with the pixel.

35. A user interface for a three-dimensional environment comprising:

a pointer, movement of which is controlled by a pointing device;

at least one graphical object displayed on at least one pixel of a display; and

a frame buffer having memory for the at least one pixel, the memory including a first field for holding display information and a second field for holding object identification information.

36. A frame buffer comprising:

a plurality of memory locations, each memory being associated with a pixel of a display device;

display data stored the memory locations, the display data indicating how the pixel is to be activated; and

object identification data stored in at least some of the memory locations, the object identification data indicating a unique software graphical object being displayed by the pixel.

37. A graphical user interface comprising:

a rendered image of at least one three-dimensional graphical object, wherein the graphical object uses a number of pixels on a display device, and wherein there is stored a respective color value for each pixel in the display device; and object identification data stored with each pixel covered by the rendered image, wherein the object identification data uniquely identifies the graphical object located at the pixel.

38. A graphical user interface comprising:

a rendered image of at least one three-dimensional graphical object, wherein the graphical object uses a number of pixels on a display device, and wherein the rendered image produced using a z-buffer technique; and

object identification data stored with each pixel covered by the rendered image, wherein the object identification data uniquely identifies the graphical object located at the pixel.

MENU
Home
Term of the Day
New Terms
Pronunciation
New Links
Quick Reference
Did You Know?
Search Tool
Tech Support
Webopedia Jobs
About Us
Link to Us
Advertising

Compare Prices: go
HardwareCentral

Talk To Us...
Submit a URL
Suggest a Term
Report an Error

Z-buffering

Last modified: Wednesday, November 07, 2001

An algorithm used in 3-D graphics to determine which objects, or parts of objects, are visible and which are hidden behind other objects. With Z-buffering, the graphics processor stores the Z-axis value of each pixel in a special area of memory called the *Z-buffer* . Different objects can have the same x- and y-coordinate values, but with different z-coordinate values. The object with the lowest z-coordinate value is in front of the other objects, and therefore that's the one that's displayed.

An alternate algorithm for hiding objects behind other objects is called *Z-sorting*. The Z-sorting algorithm simply displays all objects serially, starting with those objects furthest back (with the largest Z-axis values). The Z-sorting algorithm does not require a Z-buffer, but it is slow and does not render intersecting objects correctly.

•**E-mail this definition to a colleague**•

Related Categories

internet.com

Developer
Downloads
International
Internet Lists
Internet News
Internet Resources
IT
Linux/Open Source
Personal Technology
Small Business
Windows Technology
xSP Resources
Search internet.com
Advertise
Corporate Info
Newsletters
Tech Jobs
E-mail Offers

internet commerce

Be a Commerce Partner
Franchise Directory
Best Digital Camera
PDA Reviews
Batteries
Digital Cameras
T-Shirts
Prepaid Calling Cards
Data Recovery
Online Degrees
Home Equity Loans
Racks
Compare Prices
Web Hosting Services

*For internet.com pages about **Z-buffering CLICK HERE**. Also check out the following links!*

LINKS = Great Page!

Object-Oriented Programming

Related Terms

3-D graphics

Z-buffer

(Webopedia)
Give Us Your Feedback

Shopping
Z buffering Products
Compare Products,Prices and Stores

Shop by Category:
CD and DVD Burners
8 Model Matches

Personal CD Players
2 Model Matches

CollabNet: Get Distributed Development On Demand:		
Free Online Book: Version Control with Subversion Get the definitive book on Subversion, written for developers, by developers.	**Whitepaper:** ALM Maturity Model Learn solutions for globally distributed application lifecycle management.	**Forrester Study:** The Total Economic Impact of tl Solution Evaluate the potential f impact of the CollabNet solution

JupiterWeb networks:

internet.com | EARTHWEB | dev | graphics.com

Search JupiterWeb: Find

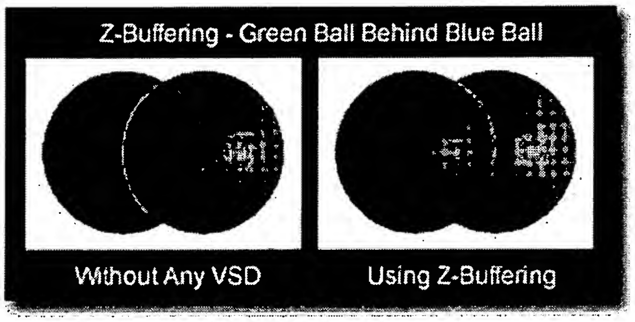
Jupitermedia Corporation has three divisions:
JupiterImages, JupiterWeb and JupiterResearch

Jupitermedia Corporate Info | Newsletters | Tech Jobs | Shopping | E-mail Offers [an error occurred while processing this directive] [an error occurred while processing this directive] [an error occurred while processing this directive] Z-Buffering

Article by : Duane Bong

Introduction To Z-Buffering

Z-buffering is a 3D imaging technique used for Visual Surface Determination [VSD]. In the real world, a solid object nearer to the viewer blocks and obscures other objects which are behind it. A quick example of this is to place your hand on the computer screen. Your hand blocks you from seeing any part of the screen that is behind it.

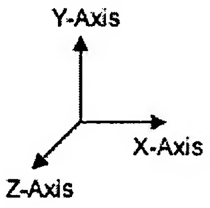


Z-Buffering Solves Overlapping Issues

However, when the same situation is generated in computer graphics, depth is lost and objects overlap each other. You will see both your hand and the image on screen at the same time! This is unavoidable because computer screens are flat and only 2 dimensional. To address this problem, Z-buffering is used to determine which objects are visible and which are hidden from view.

How Does Z-Buffering Work?

Before we proceed, let us examine a three dimensional Cartesian co-ordinate system. This has an X-axis, which defines left and right, and a Y-axis that defines up and down. It also has a Z-axis which tells us whether something will pop out of the screen or recede into it.



In Z-Buffering, the location of every pixel on the Z-axis is stored in memory. This is done for each individual object. The memory where the locations are stored is known as the Z-Buffer and this is what gives the technique its name. To decide on which objects should be displayed, the values in the Z-buffer are compared. An object with a lower Z-buffer value is considered to be in front of another object. The object with the lowest Z-Buffer value is, therefore, displayed, and the other objects hidden from view.

Z-Buffering is very good, in both static and animated scenes, at determining which objects need to be displayed. The technique's main disadvantage is that Z-Buffering needs to store and compare values of individual pixels. In many cases, this is a rather demanding load for the computer. Fortunately, most graphics cards are now equipped with specialised circuitry to speed up these demands.

Related Articles:

3D Graphics - Antialiasing Explained

3D Graphics - Texture Mapping Explained
3D Graphics - MIP Mapping Explained
3D Graphics - Alpha Blending Explained
3D Graphics - Bilinear Filtering Explained
3D Graphics - Bump Mapping Explained
3D Graphics - N-Patches Explained

Top

[an error occurred while processing this directive] [an error occurred while processing this directive]